DeepGeo: Photo Localization with Deep Neural Network

Montiel Abello mabello@andrew.cmu.edu
Nathaniel Chodosh nchodosh@andrew.cmu.edu
Sudharshan Suresh sudhars1@andrew.cmu.edu

Abstract

In this paper we address the task of determining the geographical location of an image, an extremely pertinent problem in learning and computer vision. This research was inspired from playing GeoGuessr, a game that tests a humans' ability to localize themselves using just images of their surroundings. In particular we wish to investigate how well geographical, ecological, terrain and man-made features generalize for random location prediction. This is framed as a classification problem - given images sampled from the United States, the most-probable state among 50 is predicted. Previous works use models extensively trained on large, unfiltered online datasets that are primed to specific locations. To this end, we create (and open-source) the `50States10K` dataset - with 0.5 million Google Street View images of the country. A deep neural network based on the ResNet architecture is trained, and four different strategies of incorporating low-level cardinality information are presented. This model achieves accuracies 20 times better than chance on a test dataset, which rises to 71.87% when taking the best of top-5 guesses. Results against human subjects are also presented, where the network beats humans in 4 out of 5 rounds of GeoGuessr.

1 Introduction

Games such as GeoGuessr demonstrate that humans are remarkably good at coarse estimates of location from a single image. The game presents the player with a Google Street View 360° panorama from anywhere in the world and asks the user to guess the GPS coordinates of the place the photo was captured. Humans are able to achieve decent ballpark estimates in spite of the fact that many (if not most) images have ambiguous locations unless they contain very specific landmarks.

Natural images have numerous cues that allow inference, such as vegetation, and man-made infrastructure such as roads, marking and signs. Humans are clearly able to use additional rough cues to find out where they are. Common examples of these are driving directions, languages and prior knowledge of vegetation and geography. We attempt to replicate this semantic reasoning using a deep neural network. The abundance of open-source visual data and previous success of learning methods in scene recognition motivates this effort.

Previous works, based on both learning as well as scene matching methods, have shown appreciable results at various spatial scales. We restrict the scope of the classification problem to the United States. This is an ideal playground for such a method due to its near-perfect street view coverage, high road density and varied geography. Segmenting the United States into a set of 50 classes based on state boundaries, we predict the state that an organized set of input images belongs to. In contrast to previous works which classified individual images, our goal is to train a network to be effective at geolocalization from panoramic viewpoints that are provided in GeoGuessr. Each input sample contains four images taken at the same location, facing in the cardinal directions.

More concretely, given a set of input samples $X = \{X_1, X_2, \ldots, X_n\}$ where $X_i \in \mathbb{R}^{4\times3\times256\times256}$, and a set of one-hot vector labels $Y = \{Y_1, Y_2, \ldots, Y_n\}$ where $Y_i \in \mathbb{R}^{50}$, we create a model to predict the distribution over all states that a test sample X' consists of images belonging to class $P(Y' = Y_i|X')$.

Automatic image localization is an interesting and challenging problem that tests the capabilities of deep neural networks. This tool will aid in wide-ranging applications such as meta-data aggregation, location-based priming and the creation of richer datasets. We demonstrate that deep neural networks are capable of effective geolocalization without requiring a large dataset and exhaustive training. Several approaches for integrating cardinality are employed to determine the most effective strategy for concatenation of learned features.

Our key contributions are **(a)** DeepGeo[1]: a deep neural network classifier that predicts the most-probable state a given image was captured in (not restricted to cities/landmarks) **(b)** the `50State10K` dataset, a comprehensive street view dataset of the Unites States.

2 Related Work

The challenges of geolocation based on visual information has its foundations in [1]. There have been various formulations of the same problem over the years that incorporate geotagged online photos [2, 3, 4], satellite aerial imagery [5] and city skylines [6]. Due to their extensive coverage and accuracy, Google Street View scenes have been used in [7, 8, 9]. However, these works only tackle localization at the city level, most using feature-based methods. For example [8] restricts geolocation to Pittsburgh and Orlando via SIFT descriptors and feature points.

PlaNet [3] and IM2GPS [2] are considered the baselines for scalable image geolocation over the entire world. IM2GPS computes the closest match via scene matching with a large corpus of 6 million geotagged Flickr images. They perform purely data-driven localization with features such as color and geometric information.

PlaNet [3] uses a larger such dataset paired with a deep network to obtain a probability distribution of location over a partitioned world map. It also represents the first attempt to harness a deep network to achieve coarse global localization. They formulate their task as classifying a test image to one of many partitioned multi-scale geographic cells. It uses a CNN-based on the Inception architecture, trained with 200 cores for 2.5 months.

These methods rely on sheer volume of training images - a perusal of their data yields many indoor scenes, pictures of food and pets. They are also naturally primed towards famous landmarks and cities, and are not representative of *throw a dart at a map* scenarios such as GeoGuessr. As mentioned in section 1, our method differs from the aforementioned works both in terms of classification goals and data collection. While PlaNet correctly classifies 28.4% of the test data at the country level, IM2GPS has a test set median error of 500km. It is noted that these cannot be used as comparison metrics with our method, given they attempt to tackle the problem of *global* localization while we wish to perform classification to a state in the USA. The above two methods also do not utilize Google Street View imagery to any extent. It is worth mentioning that PlaNet bested human subjects in GeoGuessr in 28 rounds of 50.

While the above two methods use large Flickr datasets, [7] uses a dataset of 100,000 GPS-tagged Street View images in two cities. To the best of our knowledge, there are no existing Google Street View datasets of either the world or the United States.

3 Dataset

3.1 Location Sampling

Rather than sampling uniformly across the United States, we sample at an equal number of locations within each state to ensure that each class is properly represented. To sample locations within each state, we use state boundary data from [10]. State boundaries are represented by a polygon connecting an ordered set of points. We sample `[latitude, longitude]` uniformly within the minimum rectangle enclosing this polygon. Points falling outside the state boundary are discarded.

The *Gridded Population of the World v4* [11] dataset is used to inform our location sampling. We use the population density map with a grid size of 2.5 arc-minutes in latitude and longitude. Within each state we normalize the population density values by dividing each grid value by the maximum

[1] https://github.com/suddhu/DeepGeo

density in the state, and discard sampled locations with a population density below a threshold p_{min}. We also ensure that duplicate locations are not sampled to ensure good coverage.

This density-based sampling allows us to achieve a good spread of locations, primes the model for more densely populated regions and boosts the hit rate of the image scraping tools described in section 3.2. For example figure 1 shows population aware sampling in Pennsylvania.

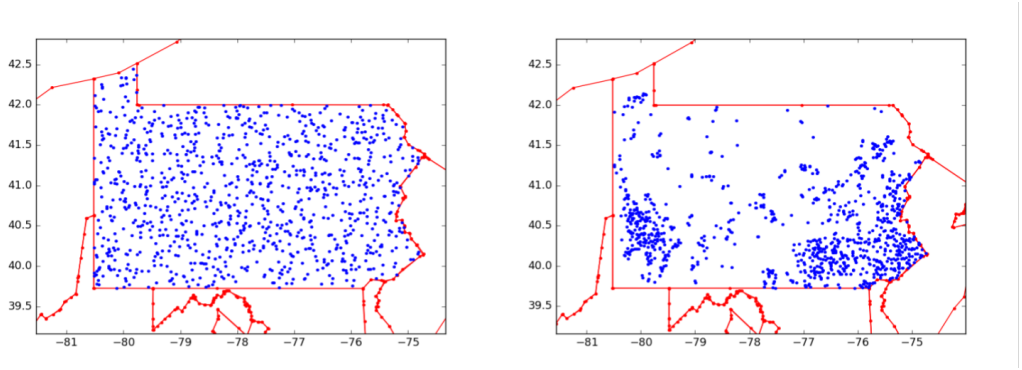

Figure 1: Sample locations in Pennsylvania for uniform and our population density aware approaches.

3.2 Data Scraping

Out dataset is obtained via the Google Street View API which provides tools for image extraction. Images were gathered over a period of 5 days, running our image scraping tools on several machines.

The sampled locations from section 3.1 are fed as queries to a script[2], which retrieves the closest unique panorama. These 360° images are split into 4 rectified images facing the cardinal directions, as shown in Figure 2. Duplicate images are pruned away to give `4 x 2500 x 50` unique natural scene images.

Figure 2: Two input samples consisting of four images each at headings of $[0°, 90°, 180°, 270°]$.

3.3 Dataset description

We present the `50States10K` dataset. To the best of our knowledge, this is the first open-source Google Street View dataset of the United States. There are 2500 samples (10000 unique images) in each of the 50 states to provide a total of 125000 samples (0.5 million images). Each sample contains 4 rectified, 256×256 Street View images taken from the sample location, facing the cardinal

[2]based on Adrian Letchford's python module.

directions. Latitude and Longitude of each sample are provided. This dataset is used to train our classifier. Figure 3 shows example images highlighting similarities and differences in features across several states.

For testing purposes, we created the `50States2K` dataset. There are 500 samples (2000 unique images) in each state, giving a total of 25000 samples. Test locations were sampled carefully to ensure uniqueness from one another and from locations in the training set.

We make the `50States10K` and `50States2K` datasets available for open source use. Each set organizes images into folders based on state location and provides the latitude and longitude corresponding to each image.

Figure 3: Sample images - California, Hawaii, North Carolina, North Dakota, and Pennsylvania datasets

4 Methods

As a baseline method we performed standard CNN classification, treating each image as a separate training example. That is we did not group multiple views from the same location in any way. The Residual Network (ResNet) is a recently published CNN architecture which we determined to be suitable for our problem[12]. The ResNet architecture uses so called *residual blocks* to achieve better gradient propagation, which enables training much deeper networks. In [12] the authors present variations of their network that have 18 to over 100 layers, we determined empirically that the 50-layer variant was sufficient. We then tested three variants of this network to determine the best way of integrating the information from multiple views of the same location. Some previous works have used sophisticated LSTM setups to accomplish this [3]. However in this case each input has a fixed number of input images, making an LSTM overly complicated. Additionally the four input images have a fixed relationship, since they always correspond to views in cardinal directions. With this in mind we chose to test simpler integration techniques described below and shown in Figure 4.

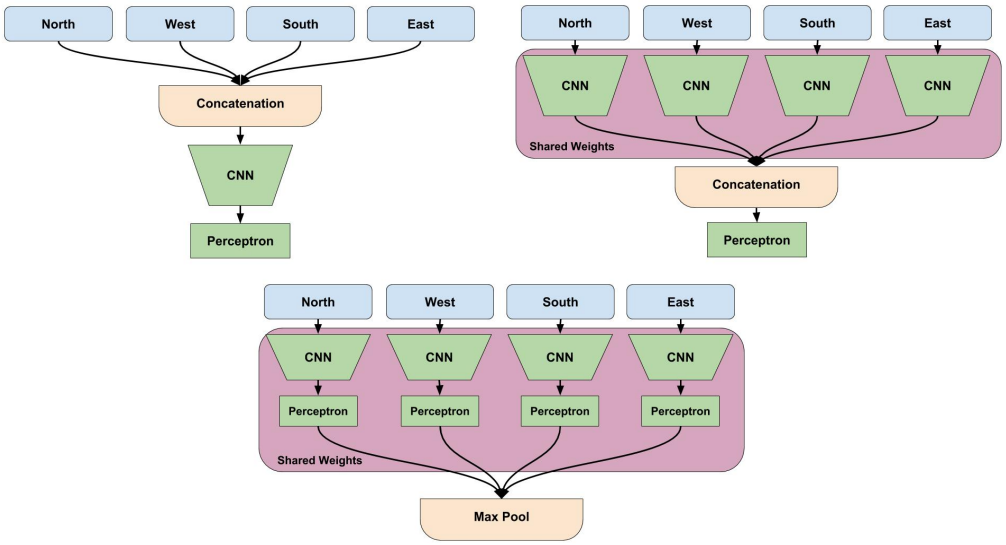

Figure 4: From left to right and top to bottom: Early Integration, Medium Integration, Late Integration

Early Integration : In our early integration scheme all four views are concatenated to form a twelve channel $(\{R, G, B\} \times \{N, S, E, W\})$ input image, allowing information to be shared between images in all layers of the network. The network is otherwise a standard 50-layer ResNet, the only modification being that the number of filters in each block of the network is doubled to account for the extra input channels.

Medium Integration : In the medium integration network we effective make the assumption that the feature extraction for each of the input images should be constant, and that integration should be done only on the extracted features. To this end, we use a CNN to get a low dimensional summary of each image, concatenate the features of each, and run a single layer perceptron on the new feature for classification. The CNN is shared across all four images, and is simply the 50-layer resnet with the fully connected layer removed.

Late Integration : For late integration we also run a shared CNN over each input image and then combine the results. However, in this scheme the CNN includes the fully connected layer and to integrate the predictions we simply take the maximum over each output class. The idea being that the only information to be gained by integration is being able to use the "best view" for the predictions, i.e the one that assigns the highest confidence to its prediction.

4.1 Implementation Details

We implemented the network in Tensorflow as described in [12] and used the Adam algorithm for training with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. We also applied weight decay with a weight of 0.0001. All three variants were trained for fifty epochs using a 90/10 train/validation split. The epoch which achieved the highest validation accuracy was then used on the test set. Training took approximately 30 hours per network running on a single Nvidia Titan X GPU.

5 Experiments and Results

5.1 Against Test Dataset

Accuracy results for the test dataset `50States2K` amongst the four different network architectures are shown below. Figure 5 depicts the average classification accuracy on the test dataset. For each architecture, we calculate the performance of the best in top-N predictions for an image. For this we sort the probability distribution of a given test image and check if the any of the greatest N values belong to the ground truth class. The rationale behind computing this metric is that some states have geographical and ecological similarities (irrespective of their proximity to each other). These are best visualized in the heatmap generated for images from Hawaii and Vermont (Figure 6).

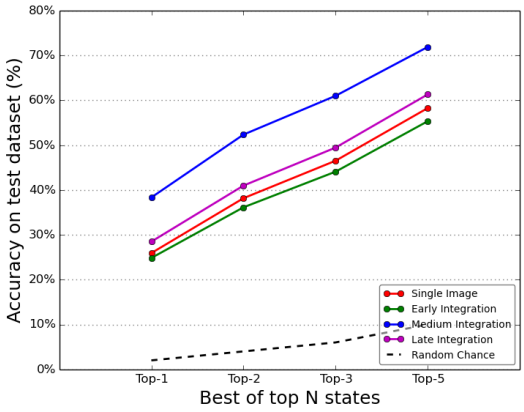

Figure 5: Best in top-N accuracy for different network architectures

Method	Top 1	Top 2	Top 3	Top 5
Single Image	25.92	38.15	46.51	58.25
Early Integration	24.83	36.12	44.07	55.32
Medium Integration	38.32	52.33	60.98	71.87
Late Integration	28.47	40.96	49.44	61.30

Table 1: Overall % accuracy of each network architecture on the test dataset `50State2K`

We see that even in the single image case, a baseline accuracy of 25.92% is obtained. The Medium Integration method yields the best results, with a 38.32% accuracy for the top-1 metric. This confirms our initial intuition that performing feature extraction on the individual cardinal directions before integration is the preferred architecture. Interestingly, the Early Integration method performs poorly, even when compared to the Single Image method.

Across the board, the accuracies improve significantly when considering the best in top-N results. This peaks at 71.87% for the Medium Integration method. PlaNet correctly classifies 28.4% of the test data at the country level. This cannot be considered to be a prior baseline for our paper, for a host of different reasons, expanded in section 2. Random chance prediction for each image are 1-in-50 or 2%. For the subsequent results, we consider those from the best-performing Medium Integration architecture.

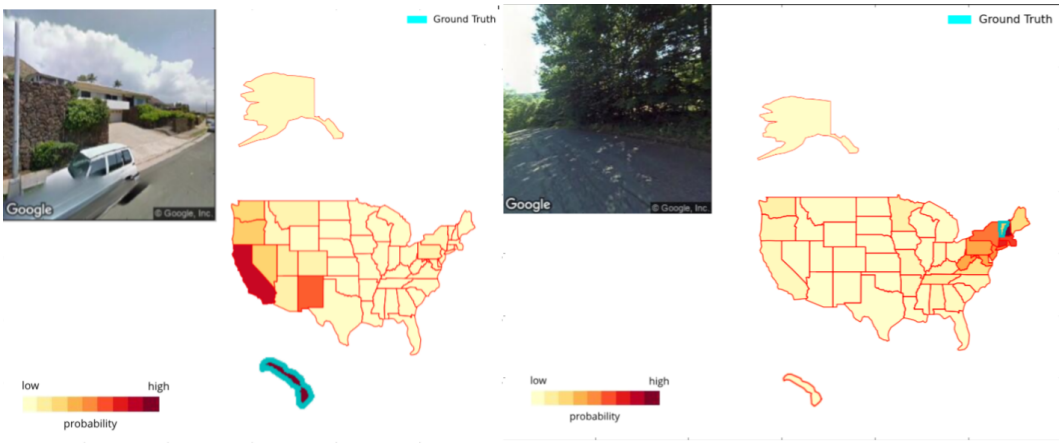

Figure 6: Heat-maps of (a) Hawaii (b) Vermont. We see that in (a) the image is also similar to those in California, while in (b) one could argue that Vermont possesses characteristics shared by other parts of the NorthEast.

We also analyze the per-class accuracies, to identify *distinctness* trends amongst states. For example, its natural to assume that a model would be able to predict images in states like Hawaii and Alaska far better than those in the contiguous USA. This is due to their disparate landscape in comparison with other states. Table 2 presents the best and worst performing states, which falls in line with our assumption. Figure 8 arranges the class accuracies of all 50 states.

Best 5		**Bottom 5**	
State	**Accuracy (%)**	**State**	**Accuracy (%)**
Hawaii	95.6	*Virginia*	9.2
Alaska	91.2	*Texas*	10.4
Arizona	73.4	*Alabama*	12.6
Wyoming	63.8	*Ohio*	14.4
Vermont	62.4	*Missouri*	16.8

Table 2: Best and worst performing classes by % accuracy

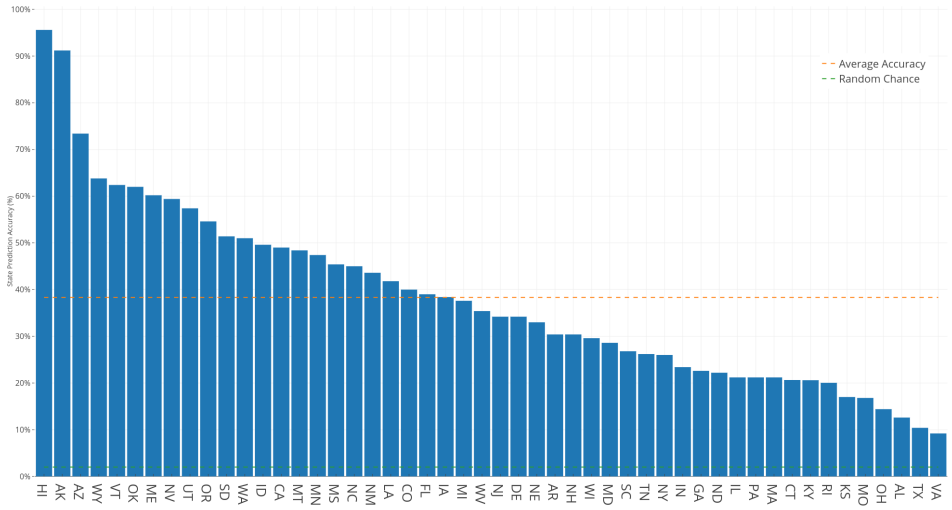

Figure 7: Accuracy among the 50 classes with the medium integration model on `50State2K`

Figure 8 plots correctly and incorrectly classified test samples. We observe that the network performs poorly on regions of low population density. As we sample based on population density due to the distribution of Street View images, the classifier is not well-trained on image features in these regions.

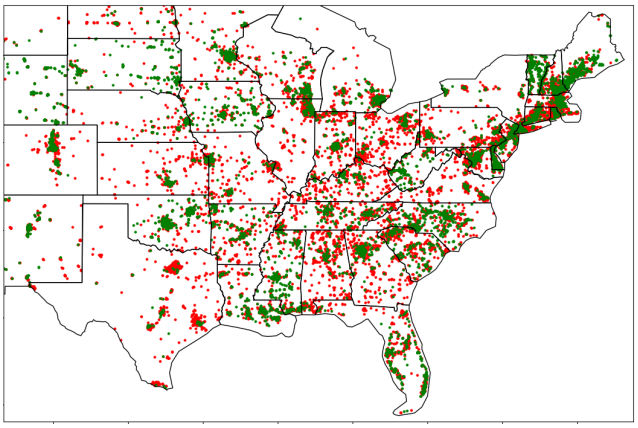

Figure 8: Test locations depicting correctly/incorrectly classified show that network performs poorly on samples from regions of low population density.

5.2 Against Humans in Geoguessr

The performance of our network was evaluated in head-to-head rounds of GeoGuessr against human subjects. 10 locations tests were performed per game. To ensure a fair test, humans were only allowed to rotate the panorama viewpoint and could not explore the map. At each location, coordinates were recorded and used to download images facing the cardinal directions for network testing. It was necessary to replay roughly 50% of the tests (with a different human player) as some locations were rejected by the Google Streetview API image downloader. Table 3 shows the results of these tests. The medium integration architecture beat humans in 4 out of 5 rounds.

Figure 9 shows representative results from 3 rounds. The leftmost image shows that mis-classifications typically arise when samples strongly resemble locations in different states. In this case, both humans and DeepGeo place guesses in the wrong region. The next two demonstrate DeepGeo outperforming the human in both large and small scale accuracy.

Game	**Human (%)**	**Early (%)**	**Medium (%)**	**Late (%)**
1	0	10	60	20
2	0	10	10	0
3	30	10	10	10
4	10	30	30	10
5	10	10	20	0

Table 3: Accuracy (%) over 10 rounds of *GeoGuessr* for human subject and early, medium and late integration architectures.

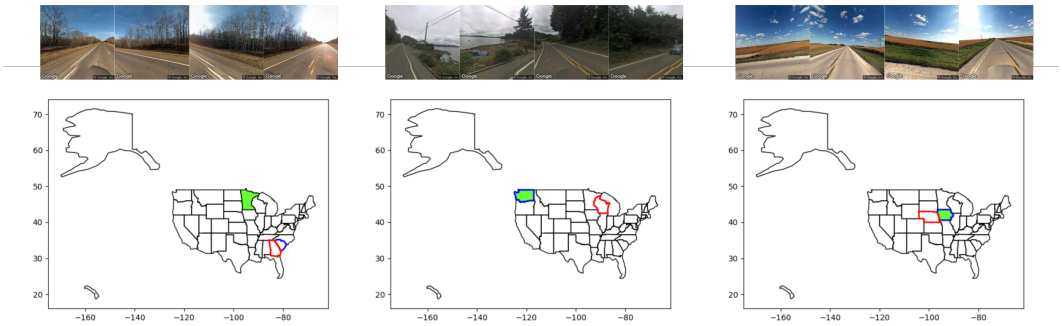

Figure 9: Correct label (green), human guess (red) and DeepGeo guess (blue) displayed on map, along with input Street View images.

6 Conclusion and Future Work

GeoGuessr poses a challenging image classification task of estimation location based on image pixels. We have presented several DeepGeo network architectures for this task, achieving excellent performance with the best variant. In addition, we open-sourced both a training and test dataset for similar tasks. We explored importance of choosing the appropriate information integration strategy for a classification task. Specifically, we have shown that in contrast to conventional wisdom, the earliest integration scheme is not always superior.

For future work, we would like to extend the model to perform regression on latitude and longitude rather than classification on state. This would also include collecting a uniformly sampled dataset, as opposed to population-based sampling. We also wish to evaluate results with different network architectures - for example, PlaNet uses the Inception architecture to obtain good results. The ideal use-case we would like to see, is when the network is able to generalize to in-the-wild photographs that don't belong to the street view dataset. We believe that this work can expand the capabilities and usefulness of the tool.

References

[1] William B Thompson, Carolyn M Valiquette, BH Bennet, and Karen T Sutherland. Geometric reasoning for map-based localization. *Computer Science Technical Report UUCS-96-006, University of Utah, Salt Lake City, UT*, 1996.

[2] James Hays and Alexei A Efros. Im2gps: estimating geographic information from a single image. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[3] Tobias Weyand, Ilya Kostrikov, and James Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016.

[4] Song Cao and Noah Snavely. Graph-based discriminative learning for location recognition. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 700–707. IEEE, 2013.

[5] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016.

[6] Srikumar Ramalingam, Sofien Bouaziz, Peter Sturm, and Matthew Brand. Skyline2gps: Localization in urban canyons using omni-skylines. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3816–3823. IEEE, 2010.

[7] Amir Roshan Zamir and Mubarak Shah. Accurate image localization based on google maps street view. In *European Conference on Computer Vision*, pages 255–268. Springer, 2010.

[8] Amir Roshan Zamir and Mubarak Shah. Image geo-localization based on multiplenearest neighbor feature matching usinggeneralized graphs. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1546–1558, 2014.

[9] Hyo Jin Kim, Enrique Dunn, and Jan-Michael Frahm. Predicting good features for image geo-localization using per-bundle vlad.

[10] Xml for us state polygons. http://econym.org.uk/gmap/states.xml. Accessed: 2018-03-30.

[11] NASA Socioeconomic Data and Applications Center (SEDAC). the gridded population of the world, version 4 (gpwv4). http://sedac.ciesin.columbia.edu/data/collection/gpw-v4/documentation, 2017. Accessed: 2018-03-30.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.